16 | 实战痛点2：项目开发中的权限系统

2021-11-22 大圣

《玩转Vue 3全家桶》 课程介绍 >

讲述：大圣

时长 09:44 大小 8.92M

你好，我是大圣。

在上一讲中，我们使用组件库完成了项目的搭建，并且引入了 Element3 和 axios 等基础库。基于 Element3，我们可以很方便地使用组件搭建项目。而使用 axios，可以很方便地获取后端数据。在项目中，权限系统的控制需要前后端配合完成，而且权限系统也是后端管理系统中常见的一个难点。不过，今天我们主要从前端的角度，来聊一下项目的权限系统。

下面，我们先从登录权限谈起，因为登录权限对于一个项目来说是必备的功能模块。完成了登录选项的设置后，下一步需要做的是管理项目中的页面权限，而角色权限在这一过程中则可以帮助我们精细化地去控制页面权限。

登录权限

继续上一讲我们搭建起来的项目，你可以看到现在所有的页面都可以直接访问了，通常来说管理系统的内部页面都需要登录之后才可以访问，比如个人中心、订单页面等等。首先，我们来设计一个这样的权限限制功能，它能保证某些页面在登录之后才能访问。

为了实现这个功能，我们首先需要模拟登录的接口和页面。我们先新增路由页面，进入到项目目录下，在 router.js 中新增路由配置。下面的代码中，routes 数组新增 /login 路由访问。

复制代码

```
import Login from '../components/Login.vue'
const routes = [
...
{
  path: '/login',
  component: Login,
  hidden: true,
}
]
```

然后，我们进入到 src/components/Login.vue 组件中，组件的代码如下所示。在代码中，我们能看到，用户在输入用户名和密码之后，把用户名和密码传递给后端，然后就可以实现登录认证。

复制代码

```
handleLogin() {
  formRef.value.validate(async valid => {
    if (valid) {
      loading.value = true
      const {code, message} = await useStore.login(loginForm)
      loading.value = false
      if(code===0){
        router.replace( toPath || '/')
      }else{
        message({
          message: '登录失败',
          type: 'error'
        })
      }
    } else {
      console.log('error submit!!')
      return false
```

```
    }
  })
}
```

由于我们的项目是一个前端项目，所以我们需要在 Vite 内部做数据结构的模拟。我们在 src 目录下面新建 mock 目录，用来放置假数据的结构。我们写死一个用户名 dasheng，使用用户名 dasheng 登录成功之后，我们把用户名、过期日期等重要信息进行加密，生成一个 token 返回给前端。

这个 token 就算是一个钥匙，对于那些需要权限才能读取到的页面数据，前端需要带上这个钥匙才能读取到数据，否则访问那些页面的时候就会显示没有权限。

复制代码

```
{
  url: '/geek-admin/user/login',
  type: 'post',
  response: config => {
    const { username } = config.body
    const token = tokens[username]

    // mock error
    if (user!=='dasheng') {
      return {
        code: 60204,
        message: 'Account and password are incorrect.'
      }
    }

    return {
      code: 20000,
      data: token
    }
  }
}
```

我们回到前端页面，登录成功后，首先需要做的事情，就是把这个 token 存储在本地存储里面，留着后续发送数据。这一步的实现比较简单，直接把 token 存储到 localStorage 中就可以了。我们拿到这个 token 后，为了进行接口权限认证，要把 token 放在 HTTP 请求的 header 内部。

我们看下面的代码，在 axios 的请求发出之前，我们在配置中使用 getToken 从 localStorage 中读取 token，放在请求的 header 里发送。由于我们使用了请求拦截的方式，所以所有的后端数据发送的时候，都会带上这个 token，完成受限数据的请求。

复制代码

```
service.interceptors.request.use(
  config => {
    const token = getToken()
    // do something before request is sent
    if (token) {
      config.headers.gtoken = token
    }
    return config
  },
  error => {
    // do something with request error
    console.log(error) // for debug
    return Promise.reject(error)
  }
)
```

通过上面的操作，我们完成了前端网络请求的 token 限制。但是还有一个需求没有实现，就是用户没有登录某个受限页面的时候，或者说没有 token 的时候，如果直接访问受限页面，比如个人中心，那么就需要让 vue-router 拦截这次页面的跳转。

与 vue-router 拦截页面的跳转，并显示无权限的报错信息相比，直接跳转登录页是现在更流行的交互方式。但这种方式需要在 vue-router 上加一层限制，这层限制就是说，在路由跳转的时候做权限认证，我们把 vue-router 的这个功能称作导航守卫。

关于导航守卫的 API，你可以从 Vue Router 的官网看到很详细的介绍。这里我们实际应用下，下面的代码中，我们在 router.beforeEach 函数中设置一个全局的守卫。

每次页面跳转的时候，vue-router 会自动执行守卫函数，如果函数返回 false 的话，页面跳转就会失败。不过，我们也可以手动地跳转到其他页面。现在我们设置的路由很简单，如果 token 不存在的话直接跳转登录页面，否则返回 true，页面正常跳转。

复制代码

```
router.beforeEach(async (to, from,next) => {
  // canUserAccess() 返回 `true` 或 `false`
```

```
  let token = getToken()
  if(!token){
     next('/login')
  }
  return true
})
```

当然，在路由守卫的函数内，只要是页面跳转时想实现的操作，都可以放在这个函数内部实现，比如一些常见的交互效果，就像给项目的主页面顶部设置一个页面跳转的进度条、设置和修改页面标题等等。和我们现在对全部页面进行一次性的粗略拦截相比，后面还会在路由守卫那里进行更精确的路由拦截。

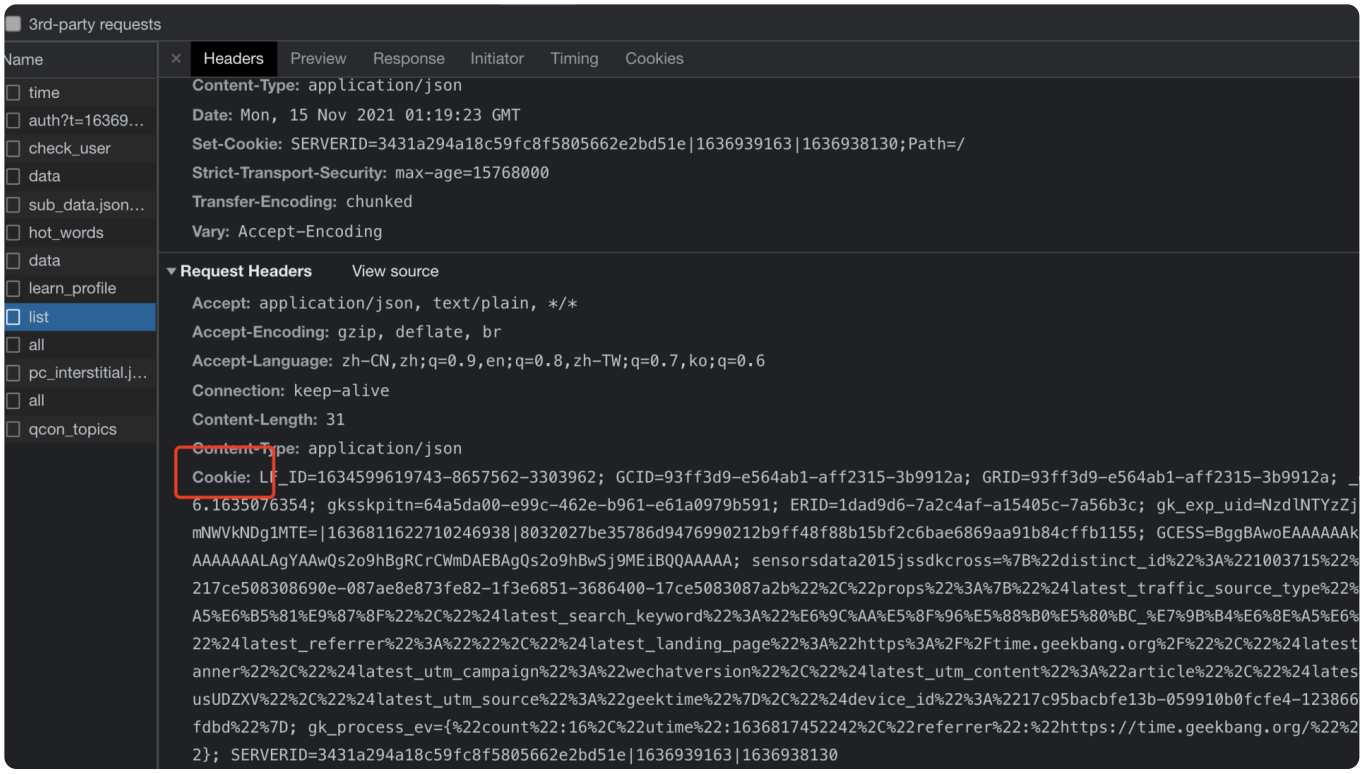
到这里你可能会有疑问：之前开发项目的时候，和登录注册相关的配置，不需要自己管理 token，都是后端直接设置 cookie。那么这里用到的 token 和之前项目开发时，交给后端设置的 cookie 到底有什么区别呢？

这是个非常好的问题，我们在第一讲聊前端发展史的时候，提到了 jQuery 时代的前端项目是作为后端项目的模块部署的。

那时候前后端不分家，整个应用的入口是后端控制模板的渲染。在模板渲染前，后端会直接判断路由的权限来决定是否跳转。登录的时候，后端只需要设置 setCookie 这个 header，之后浏览器会自动把 cookie 写入到我们的浏览器存起来，然后当前域名在发送请求的时候都会自动带上这个 cookie。

在 Chrome 浏览器中，我们先进入极客时间的官网，然后打开调试窗口页面，再选择 Network 页面。之后，我们在页面中点击 Fetch/XHR，然后在 Name 这一栏中，我们可以任选一个接口点开。这样，我们就可以看到这个接口请求的所有细节了。

在下图中，我们点击 list 请求，也就是极客时间的推荐接口时，HTTP 的 Request Headers 里就有 Cookie 这个数据，这是浏览器自动管理和发送的，也算是权限认证的最佳方案之一。



但是，在现在这种前后端分离的场景下，通常前后端项目都会部署在不同的机器和服务器之上，Cookie 在跨域上有诸多的限制。所以在这种场景下，我们更愿意手动地去管理权限，于是就诞生了现在流行的基于 token 的权限解决方案，你也可以把 token 理解为我们手动管理的 cookie。

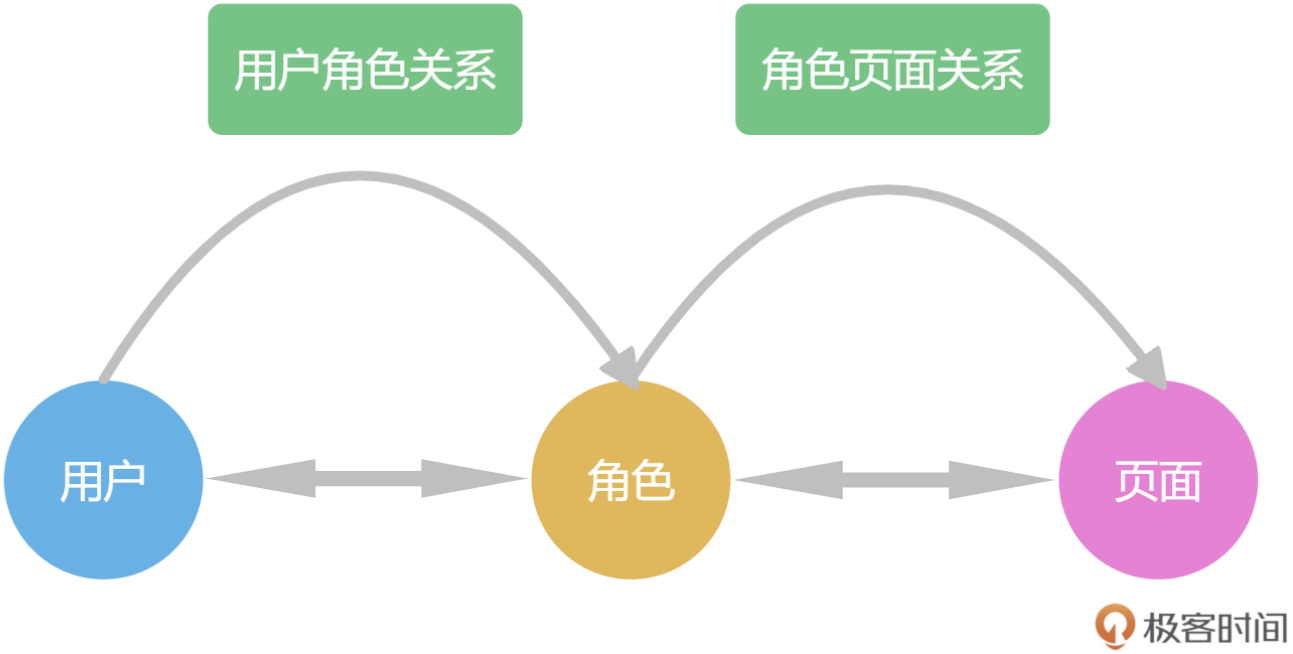
角色权限

实现登录权限验证之后，我们就可以针对项目中的页面进行登录的保护。但现在，我们只能通过登录状态去判断页面的显示与否，而这远远达不到我们实际开发的需求。

比如，在我们的管理系统开发中，订单页面是所有人都可以看到的，但是像账单的查询页面，以及其他一些权限更高的页面，我们需要管理员权限才能看到。这时候，我们就需要对系统内部的权限进行分级，每个级别都对应着可以访问的不同页面。

我们通常使用的权限解决方案就是 RBAC 权限管理机制。简单来说，就是在下图所示的这个模型里，除了用户和页面之外，**我们需要一个新的概念，就是角色。**每个用户有不同的角色，每个角色对应不同的页面权限，这个数据结构的关系设计主要是由后端来实现。

根据下图这个结构，在用户登录完成之后我们会获取页面的权限数据，也就是说后端会返回给我们当前页面的动态权限部分。



这样有一部分页面是写在代码的 src/router/index.js 中，另外一部分页面我们通过 axios 获取数据后，通过调用 vue-router 的 addRoute 方法动态添加进项目整体的路由配置中。

关于这部分动态路由的内容，官网的文档中有详细的API 介绍。在下面的代码中，我们在 Vuex 中注册 addRoute 这个 action，通过后端返回的权限页面数据，调用 router.addRoute 新增路由。

复制代码

```
addRoutes({ commit }, accessRoutes) {
  // 添加动态路由，同时保存移除函数，将来如果需要重置路由可以用到它们
  const removeRoutes = []
  accessRoutes.forEach(route => {
    const removeRoute = router.addRoute(route)
    removeRoutes.push(removeRoute)
  })
  commit('SET_REMOVE_ROUTES', removeRoutes)
},
```

与新增路由对应，在页面重新设置权限的时候，我们需要用 router.removeRoute 来删除注册的路由，这也是上面的代码中我们还有一个 remoteRoutes 来管理动态路由的原因。

然后，我们需要把动态路由的状态存储在本地存储里，否则刷新页面之后，动态的路由部分就会被清空，页面就会显示 404 报错。我们需要在 localStorage 中把静态路由和动态

路由分开对待，在页面刷新的时候，通过 src/router/index.js 入口文件中的 routes 配置，从 localStorage 中获取完整的路由信息，并且新增到 vue-router 中，才能加载完整的路由。

权限系统中还有一个常见的问题，就是登录是有时间限制的。在常见的登录状态下，token 有效期只能保持 24 小时或者 72 小时，过了这个期限，token 会自动失效。即使我们依然存在 token，刷新页面后也会跳转到登录页。所以，对 token 有效期的判断这个需求该如何实现呢？

首先，token 的过期时间认证是由后端来实现和完成的。如果登录状态过期，那么会有一个单独的报错信息，我们需要在接口拦截函数中，统一对接口的响应结果进行拦截。如果报错信息显示的是登录过期，我们需要清理所有的 token 和页面权限数据，并且跳转到登录页面。

下面的代码中，50008 和 50012 都代表着状态过期，我们可以直接使用 Element3 的 messageBox 组件显示一个错误信息，提示用户需要重新登录，然后直接跳转到登录页面就可以了。

复制代码

```
// 50008: Illegal token; 50012: Other clients logged in; 50014: Token expired;
if (res.code === 50008 || res.code === 50012) {
  // to re-login
  Msgbox.confirm('You have been logged out, you can cancel to stay on this pag
    confirmButtonText: 'Re-Login',
    cancelButtonText: 'Cancel',
    type: 'warning',
  }).then(() => {
    store.dispatch('user/resetToken').then(() => {
      location.reload()
    })
  })
}
return Promise.reject(new Error(res.message || 'Error'))
```

总结

今天这一讲的内容就讲完了，我们来总结一下今天学到的重点吧。首先，在登录权限这一部分，我们模拟了登录的接口和页面，登录成功后的 token 存储在 localStorage 中，使用 axios 的拦截器管理所有接口的 token 信息。

然后，我们学习了路由守卫的概念，借助路由守卫，我们可以实现页面的权限保护，让指定页面需要登录之后才可以访问。我们使用的是 token 权限校验机制，在用户登录之后，后端会把用户信息加密成一个 token 返回给前端，然后前端把 token 存储在 localStorage 中。

在 axios 的全局拦截函数中，每个请求发出去之前都会获取 token，然后放在 HTTP 请求的 header 之中，后端会读取这个 token 进行权限认证。除了后端接口的权限，vue-router 也会做页面的权限拦截，这个功能也就是路由守卫，vue-router 允许你在所有路由跳转之前执行钩子函数，这个函数内部我们做权限认证，如果符合要求就允许跳转，否则就跳转到 404 页面。

最后，我们还了解了更复杂一些的权限系统设计，在这种设计下，用户可以根据角色的划分，对应到不同的页面权限，一部分页面是登录后就可以访问，一部分页面是需要额外地获取后端接口，匹配到当前用户权限之后，才可以访问的页面，我们把这种权限设计称之为动态路由。vue-router 提供的 addRoute 和 removeRoute 这两个函数，可以很好地帮助我们实现这一功能。

思考题

最后，是给你留的一个思考题，如果在任意一个页面里，我们想实现按钮级别的权限认证，那我们应该如何扩展我们的权限系统呢？

欢迎在留言区分享你的看法，也欢迎你把这一讲的内容分享给你的同事、朋友们，我们下一讲再见。

分享给需要的人，Ta订阅后你可得 20 元现金奖励

生成海报并分享

赞 8 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 15 | 实战痛点1：复杂 Vue 项目的规范和基础库封装

下一篇 17 | 实战痛点3：Vue 3 中如何集成第三方框架

更多课程推荐

精选留言 (20)

写留言

一个自闭的人
2021-11-22

老师能否把代码贴出来呢，这样比较直观点

展开

共 4 条评论 > 7

II 民
2021-11-22

大圣讲的很详细，点赞，我在课后还是要实际写遍代码，跑起项目来才能更好的吸收知识。

另外关于登录状态有个疑问，网上搜索了下，不确定是否叫做，多点登录状态限制问题。就是像极客时间这种网页登录后，在别的设备登录后，这边就下线了。...

展开

共 3 条评论 > 6

酱汁
2021-11-22

封装自定义指令 v-auth 可以精确到按钮级别的权限

展开

6

.K
2021-11-24

使用老师之前的代码按照文章的流程实现：
1. mock数据
2. jwt生成Token验证
3. 拦截器Token加头
4. 路由守卫权限验证...

展开

共 1 条评论 >

3

Geek_liuke
2021-11-22

做项目的时候遇到过大厂后台不愿意做token过期判断的情况。只有前端自己设置token过期时间和判断，需要在axios里增加一个每成功调用一次接口就自动延长token期限的逻辑。在页面中还需要增加一个watch，监听token期限的变化，期限到了或者token被清空了就自动跳转到登录页去。

展开

作者回复: 我做过类似的需求，是对接了三个系统，三个token的过期时间不一样

3

海阔天空
2021-11-22

权限系统确实是实战疼点。token 的过期时间认证是由后端来实现和完成的。所以我们需要做http请求的响应拦截。以此判断是否登录和是否登录过期。请问老师，如果做了响应拦截，是否还需要路由守卫呢？如果使用路由守卫？

展开

共 3 条评论 >

3

M.
2021-11-28

每一讲应该附带这一讲对应的整体代码结构，应该更容易了解！！

2

一块小砖头
2021-11-25

我还是希望老师能结合视频一起来，这个样缺胳膊少腿容易。

1

ohmylittleego .
2021-11-25

大圣，实战篇的代码 Git上还没出吗？

展开

共 1 条评论 >

1

Condor Hero
2021-11-22

自定义组件和自定义指令都行，更推荐自定义组件来实现。

1

清歌终南
2021-11-22

使用Vue中的自定义指令，加入按钮权限

展开

1

小胖
2021-11-22

我们项目按键权限的实现是，后端返回所有用户权限信息，直接v-if权限表里面有没有这个字段(这中间匹配有个方法)。
同时，后端也会对按钮对应的接口在权限上添加上限制。

展开

1

南山
2021-11-22

封装v-auth指令，控制按钮显示，老师能讲下refreshToken的处理吗？

共 4 条评论 >

1

。。。
2021-12-02

希望结合一个项目从零到运用来结合讲解更能理解 这样太散了

编辑回复: 后面组件设计部分会带着大家用代码体验一下。

葱味黑咖啡
2021-12-01

"在页面刷新的时候，通过 src/router/index.js 入口文件中的 routes 配置，从 localStorage 中获取完整的路由信息"。
老师，有一个问题，路由对象的component可以存储在localStorage中吗？localStorage应该是只能存字符串吧，把路由信息json化之后，component也无效了，这个怎么解决

展开

阿阳
2021-11-29

请问老师，在登录时这条语句 const {code, message} = await useStore.login(loginForm)，能解释详细一点嘛，不理解useStore.login()这种用法。

展开

666de6
2021-11-29

实现按钮级别的权限认证:
1. 维护页面下需要控制权限的按钮权限标识，后台保存；
2. 登录后，获取权限数据，将该用户的按钮权限数组存放到对应页面的路由信息里；
3. 可编写v-auth的自定义指令（可以拿当前按钮标识去当前页面路由信息的按钮权限数组里去找，存在则显示，否则隐藏）；

展开

Jace
2021-11-24

大圣老师，能不能搞一个完整的项目，然后再结合知识点讲解下，这样效果好点

共 2 条评论

Danny
2021-11-24

Mock那块需要改动vite 的配置吗？

展开

淡若清风过

2021-11-22

属性(property) Ref<null>.value: null这个怎么解决？

```
const p = ref(null)

onMounted(() => {
  p.value....
```

展开